



Preparação da Game Engine para a Prática

Já revemos muitos conceitos de Game Design, Engenharia de Software, Design Gráfico, C# e agora vamos colocar a mão na massa para fazer um jogo. Para tanto será necessário você fazer uso do Unity, Inkscape e muita criatividade.

Você poderá usar nossa máquina virtual para desenvolver esse projeto ou instalar esses softwares em seu próprio computador. Iremos começar o desenvolvimento desse jogo usando alguns assets criados por mim, e pacotes gratuitos da própria Unity. Lembre-se: para a comercialização do jogo, você precisaria fazer todos ou então comprar os assets.

Iremos criar um jogo inspirado no Angry Bird, porém, com a temática da guerra de Troia. As ilustrações já foram feitas no estilo flat e irei disponibilizar para vocês.

Para iniciarmos precisaremos percorrer a seguinte trilha:



- **Preparação de Assets:** aprender como preparar assets para colocá-los em Prefers;



- **Input touch phase e count:** aprender a fazer que o jogo reconheça o toque na tela do smartphone;




Input touch Position e Accelerometer: aprender a fazer o movimento dos dedos sobre a tela e/ou do próprio celular responder a um script;

- **Line Renderer:** saber o que é e como criá-la.

- **Preparar os Assets**

Antes de mais nada, você precisará criar um projeto na Unity para o desenvolvimento de nosso jogo 2D. Depois, para carregar um arquivo de imagem na pasta do projeto, podemos usar a barra de pesquisa para encontrar o ativo.

Será muito importante para continuarmos nossas aulas que você tenha exportado os assets criados por você, ou ainda ter feito o download dos arquivos que irei disponibilizar em materiais de apoio importando-os (Figura 1) para a Game Engine (NASCIMENTO, 2022).



Figura 1 – Importando Assets para a Unity | Fonte: Autor, 2022.

Depois será importante criar Prefabs, ou seja, GameObjects pré-configurados que você cria na cena e armazena no seu projeto. Para tanto, vá em p[illegible]os na pasta assets e crie uma pasta chamada "Prefabs/" (Figura 2).

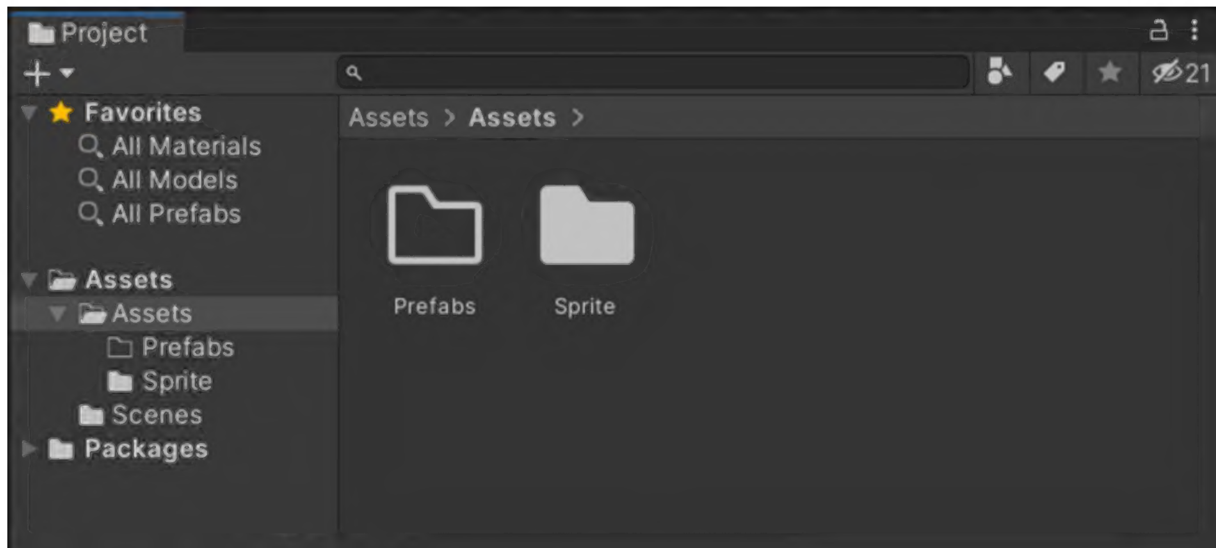


Figura 2 – Criação da pasta para os Prefabs. | Fonte: Autor, 2022.

As configurações dos Prefabs poderão variar de acordo com o objeto e mecânica que ele terá no jogo. Em atividade prática explicarei passo a passo a configuração de cada objeto. Lembrando que entre os objetos teremos (Figura 3):

- **Spartano:** Personagem que será arremessado sobre as muralhas, para tanto, precisará editar a sua física para ter um "Circle Collider 2D" e "Rigibody 2D";

- **Cenário:**

- **Montanhas:** que ficarão de fundo;

-

Vale: base do primeiro plano que representará o chão;



- **Muralhas e torres:** nesse caso precisaremos editá-las para ter corpos físicos "Box Collider 2D" e "Rigibody 2D";

- **Nuvens:** para animar o céu;

- **Pedras-Frente:** será usada no primeiríssimo plano para gerar efeito paralax.

- **Catapulta:** que substituirá o estilingue usado no jogo inspiração usado para lançar personagem contra a muralha, para esse item, precisaremos editar o local que se encontra o ponto pivô.



Figura 3 – Sprites | Fonte: Autor, 2022.

Agora que você conseguiu organizar as sprites gostaria de dar a dica para vocês usarem o Unity Remote para verificar os resultados do que você está programando. Com o Unity Remote, você pode usar um dispositivo Android e/ou iOs para visualizar e testar seu jogo ao vivo, diretamente no Unity Editor 5.4 ou posterior.

Segundo a loja de aplicativo do Google (2022):

Unity Remote 5 faz com que seu dispositivo Android funcione como controle remoto. Ele transmite eventos de mudança de toque, acelerômetro, giroscópio, webcam e orientação de tela de volta para o Unity Editor. Isso é útil para o desenvolvimento rápido quando você não deseja compilar e implantar seu projeto no dispositivo Android para cada alteração e prefere usar a janela Unity Editor Game.

2. **Input touch phase e count:**

Segundo Pereira et al. (2021, p. 110), a maioria dos jogos usam a entrada de informações do usuário para realizar ações do jogo, como dados multitoque ou acelerômetros em dispositivos móveis. Para implementar as buscas por essas ações, o Unity usa a classe Input para gerenciar eixos (mouse, joystick) e botões (teclado), ou seja, saber se o jogador está pressionando um botão ou tecla.

O primeiro código que iremos fazer é o Touch Phase, para tanto você terá que criar seu primeiro arquivo de código C# e nomeá-lo com o nome "Phase".

Primeiro você precisará colocar uma condicionante sobre o toque na tela do smartphone. O Input Touch Phase (Figura 4) irá criar a possibilidade do jogo interagir com o toque do jogador na tela do celular. Para tanto é necessário

pensar em como refletir as ações possíveis do jogador com o touch do smartphone, por exemplo:



- Tocar (TouchPhase.Began);
- Tirar o dedo (TouchPhase.Ended);
- Mover o dedo (TouchPhase.Moved);
- Mover o celular da horizontal para a vertical (TouchPhase.Canceled);
- Apertar a tela e segurar (TouchPhase.Stationary).

E:\Descomplica\Prática Integradora Game Engines\Troia\Troia\Assets\Scripts\phase.cs • - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

phase.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class phase : MonoBehaviour {

    public Text txt;
    public Touch toque;


    void Update()
    {
        if (Input.touchCount > 0)
        {
            toque = Input.GetTouch (0);

            switch (toque.phase)
            {
                case TouchPhase.Began:
                txt.text = "Began";
                break;
                case TouchPhase.Ended:
                txt.text = "Ended";
                break;
                case TouchPhase.Moved:
                txt.text = "Moved";
                break;
                case TouchPhase.Stationary:
                txt.text = "Began";
                break;
                case TouchPhase.Canceled:
                txt.text = "Canceled";
                break;
            }
        }

    }
}
```

Figura 4 – Código C# - Input Touch Phase | Fonte: Autor, 2022.

Agora é a vez de aprendermos o Input Touch Count (Figura 5), ou seja, o número de toques sobre a tela. Esse código fará que quando você toque com o dedo na tela do smartphone o jogo reconheça a quantidade de dedos que estão tocando a tela.

```
E:\Descomplica\Prática Integradora Game Engines\Troia\troia-game\Assets\Scripts\tCount.cs - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
position.cs    tCount.cs    phase.cs

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class tCount : MonoBehaviour {

    public Text txt;

    void Update()
    {
        if (Input.touchCount > 0)
        {
            txt.text = Input.touchCount.ToString ();
        }

    }
}
```

Figura 5 – Código C# - Input Touch Count | Fonte: Autor, 2022.

3. **Input touch Position e Accelerometer**

Continuando com os Inputs, iremos falar sobre o Touch Position (Figura 6), ou seja, o reconhecimento da posição onde o jogador está tocando na tela. O código consiste em dividir a tela em duas partes, com isso, se você tocar na parte que vale 2 tocará à direita, se não à esquerda.

E:\Descomplica\Prática Integradora Game Engines\Troia\troia-game\Assets\Scripts\position.cs • - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

position.cs phase.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class position : MonoBehaviour{

    public Text txt;
    Touch toque;

    void Update() {
        if (Input.touchCount > 0)
        {
            toque = Input.GetTouch (0);

            if (toque.phase == TouchPhase.Began)
            {
                if (toque.position.x > (Screen.width / 2))
                {
                    txt.text = "Direita";
                }
                else {txt.text = "Esquerda";}
            }
        }
    }
}
```

Figura 6 – Código C# - Input Touch Position | Fonte: Autor, 2022.

Já no Input Accelerometer (Figura 7) tem como objetivo fazer com que o movimento do seu smartphone seja reconhecido como meio de controle do jogo.

E:\Descomplica\Prática Integradora Game Engines\Troia\Troia\Assets\Scripts\accelerometer.cs • - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

accelerometer.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class accelerometer : MonoBehaviour {

    void Update()
    {
        transform.Translate (Input.acceleration.x, Input.acceleration.y, 0);

    }
}
```

Figura 7 – Código C# - Input Accelerometer | Fonte: Autor, 2022.



Veja a seguir alguns Inputs que poderá usar para o desenvolvimento de seu jogo:

- **Input.GetAxis:** que retorna valores numéricos de eixos de posição do mouse, teclas de movimentação (A, W, S D, Left, Up, Right e Down) e combinações das teclas Ctrl, Alt e Cmd do teclado;
- **Input GetButton:** aceitam somente parâmetro String com o nome do botão ou do Axis;
- **Input GetKey:** aceitam somente parâmetros String com o nome do botão ou um objeto KeyCode;
- **Input GetMouseButton:** recebem um parâmetro com os botões do mouse representados por números inteiros.

É claro que existem vários outros que você pode buscar em tutoriais ou guias na internet e no site do próprio Unity e Microsoft.

Agora vamos ver como iremos fazer uma linha usando o render no Unity.

4. Line Renderer

O componente Line Renderer pega uma matriz de dois ou mais pontos no espaço 2D e/ou 3D – o que os diferencia é a quantidade de eixos utilizados para o desenvolvimento do objeto – e desenha uma linha entre cada ponto. Você pode usar o Line Renderer para desenhar qualquer coisa, desde simples linhas retas até espirais complexas. Essa linha é sempre contínua; se você precisar desenhar duas ou mais linhas completamente independentes, deverá usar vários objetos de jogo, cada um com seu próprio renderizador de linha.

Vamos entender como é o artifício Line Renderer, ele será um componente muito importante para nosso game.

Você encontrará esse componente em “Component>Effects>Line Renderes”, você poderá utilizá-lo para diversos objetivos, por exemplo, usando-o para fazer uma flecha, fazer o efeito elástico etc.

O Line Render será responsável pela parte visual, lógico que para a Line tenha o efeito de elástico que queremos para o nosso jogo, precisaremos criar um efeito físico, para tanto, acrescentaremos a ele scripts e mais algumas outras configurações.

Para você criar o Line Render é necessário criar um GameObject vazio. Usando o Inspector defina esse GameObject como um ponto pequeno, nomeio para que você saiba que ele será o ponto de origem da sua Line, eu o nomearei como “P1”. Agora você precisará duplicar o ponto colocando-o em outro lugar, nomeie-o como “P2”. Depois é só ir em “Add Component” no Inspector e adicionar em Effects a Line Render, com isso você abrirá um leque de configurações no Inspector.

Em atividades práticas veremos o passo-a-passo de como usá-lo em nosso game. Espero que tenha gostado.

Atividade Extra

Leia o E-book:

REINOSO, L. F.; TEIXEIRA, G. F.; RIOS, R. O. **Jogos digitais : princípios, conceitos e práticas.** Vitória, ES: Edifes, 2020. Disponível em: <https://repositorio.ifes.edu.br/handle/123456789/812>.

Apresentação do Livro:



Este material foi criado como um recurso de nível profissional e acadêmico, expondo metodologias, conceitos e prática para desenvolver um jogo digital. O livro foi produto de experiências, leituras e referências importantes, portanto, é crucial que você leia todas as notas e dicas, faça as atividades e siga as instruções criadas, o intuito é fornecer formação suficiente para sua independência criativa. O objetivo principal é introduzir os leitores no desenvolvimento de jogos, dando capacitação mínima para compreender o universo dos games (Jogos digitais/eletrônicos).

Referência Bibliográfica

GOOGLE. **Unity Remote 5.** Califórnia: Google Play, 2022. Disponível em: <https://play.google.com/store/apps/details?id=com.unity3d.mobileremote&hl=pt_BR&gl=US>.

NASCIMENTO, W. **Jogos 2D com Unity e C#.** São Paulo: Udemy, 2022.

PEREIRA, A. I. S.; RIBEIRO, F. A. A.; CARNEIRO, P. L. S.; ARAUJO, R. C. **Desenvolvimento Jogos com Unity: Praticando do básico ao avançado.** Amazon, 2021. Edição do Kindle.

Atividade Prática: Preparação da Game Engine para a Prática



Título da Prática: Preparação da Game Engine para a Prática

Objetivos: Importar Sprites, organizá-las com Componentes e salvá-las como Prefabs.

Materiais, Métodos e Ferramentas: Para realizar esta prática vamos utilizar o software Unity, um smartphone Android e acesso ao Youtube.

Atividade Prática

Roteiro

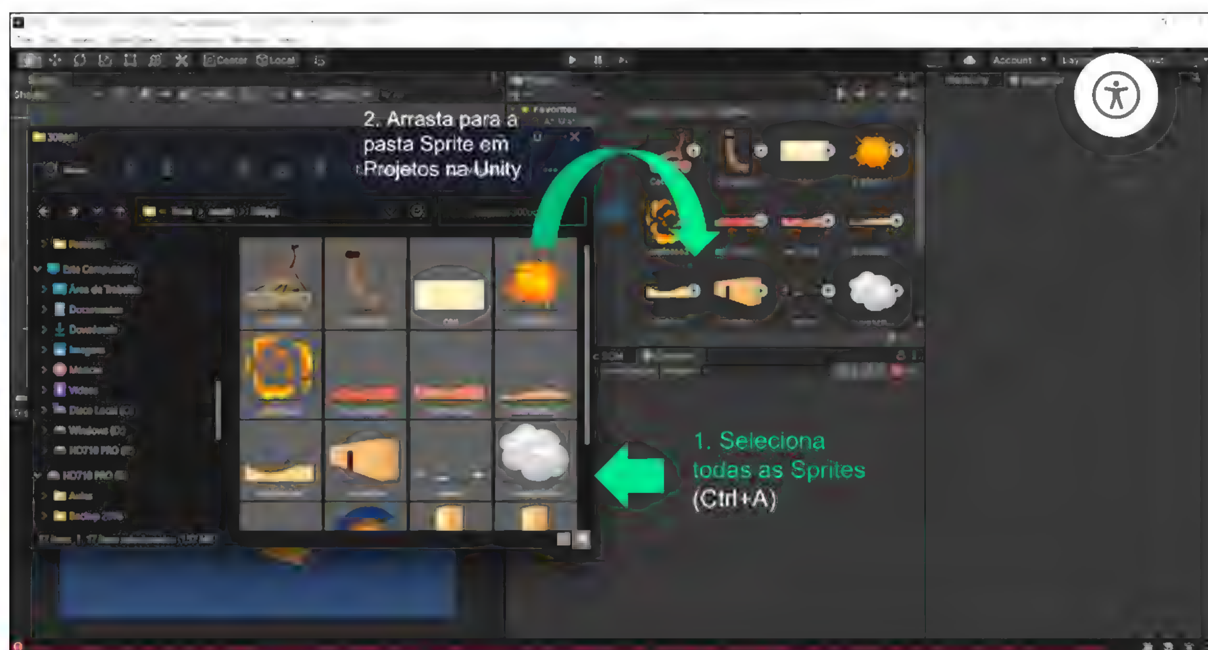
1º Passo – Leia o texto do material de apoio.

2º Passo – Assista aos vídeos da disciplina e aula 13.

3º Passo – Baixe os Assets do Game no link: https://bit.ly/3525dCT.

4º Passo – Importe as sprites para o Unity, arrastando de uma janela com a pasta onde contém os arquivos e arrastando para dentro do Project na pasta de Sprites (Figura 1).

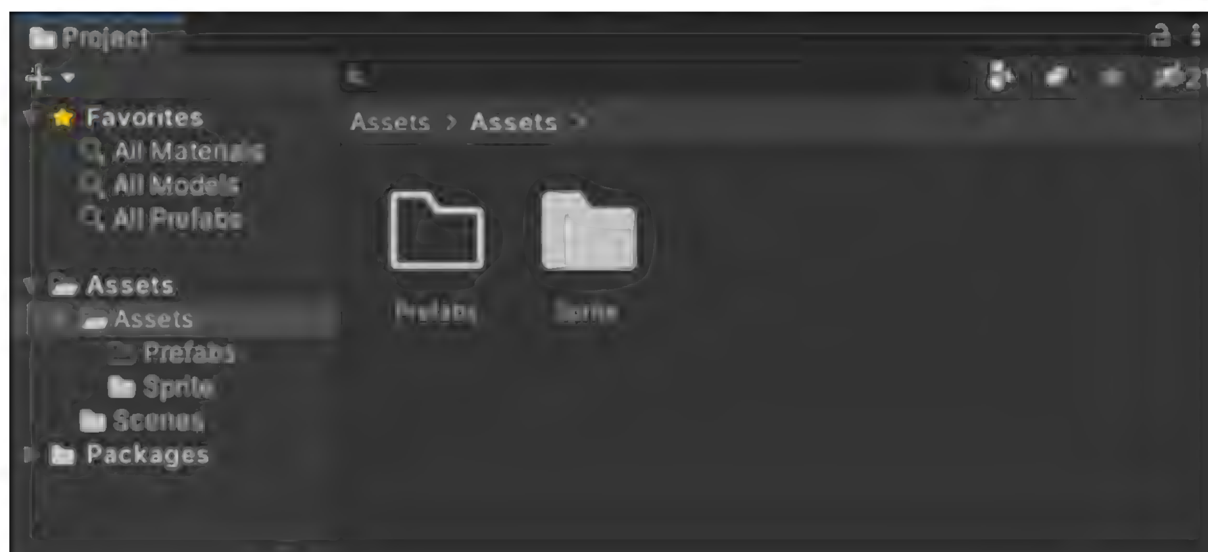
Figura 1 – Importando Assets para a Unity



Fonte: Autor, 2022.

5º Passo – Crie uma pasta para os Prefabs, ou seja, GameObjects pré-configurados que você cria na cena e armazena no seu projeto (Figura 2).

Figura 2 – Criação da pasta para os Prefabs.

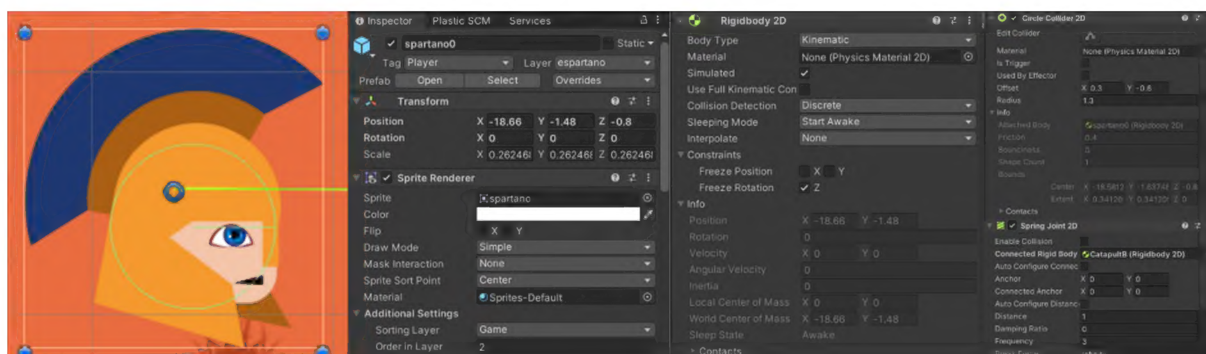


Fonte: Autor, 2022.

6º Passo – Configure os Prefabs das seguintes formas:

a) Spartano: Personagem que será arremessado sobre as muralhas, para tanto, precisará editar a sua física para ter um “Circle Collider 2D” e “Rigidbody 2D” (Figura 3);

Figura 3 – Prefab Spartano



Fonte: Autor, 2022.

b) Cenário:

Figura 4 – Prefab Muralha

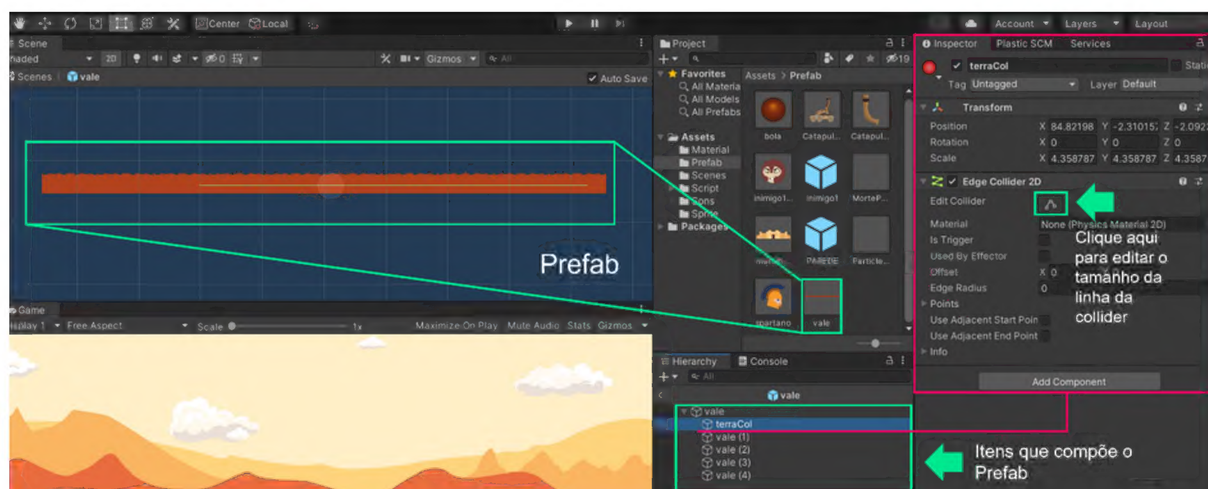
- **Muralhas e torres:** nesse caso precisaremos editá-las para ter corpos físicos “Box Collider 2D” e “Rigibody 2D” (Figura 4);



Fonte: Autor, 2022.

- **Vale:** será o chão, portanto, será necessário vinculá-lo a um tipo de Edge Collider 2D (Figura 5)





Fonte: Autor, 2022.

c) Catapulta: que substituirá o estilingue usado no jogo inspiração usado para lançar personagem contra a muralha, para esse item, precisaremos editar o local que se encontra o ponto pivô (Figura 6).

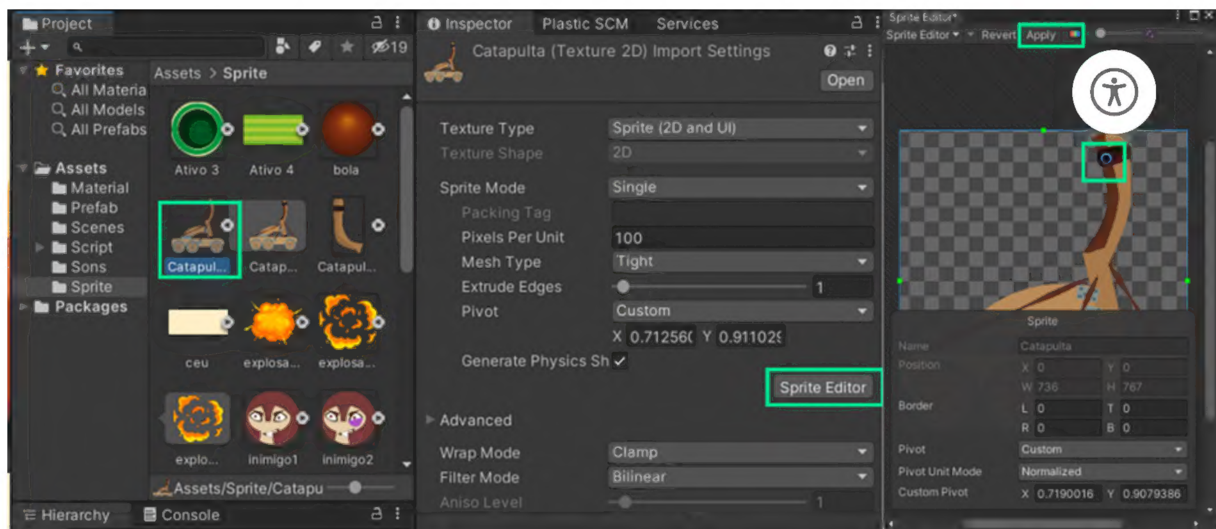
I) Clique na Catapulta;

II) Clique em Sprite Editor;

III) Mude o ponto pivô para onde quer que nasça a Line Renderer.

IV) Aperte em Apply.

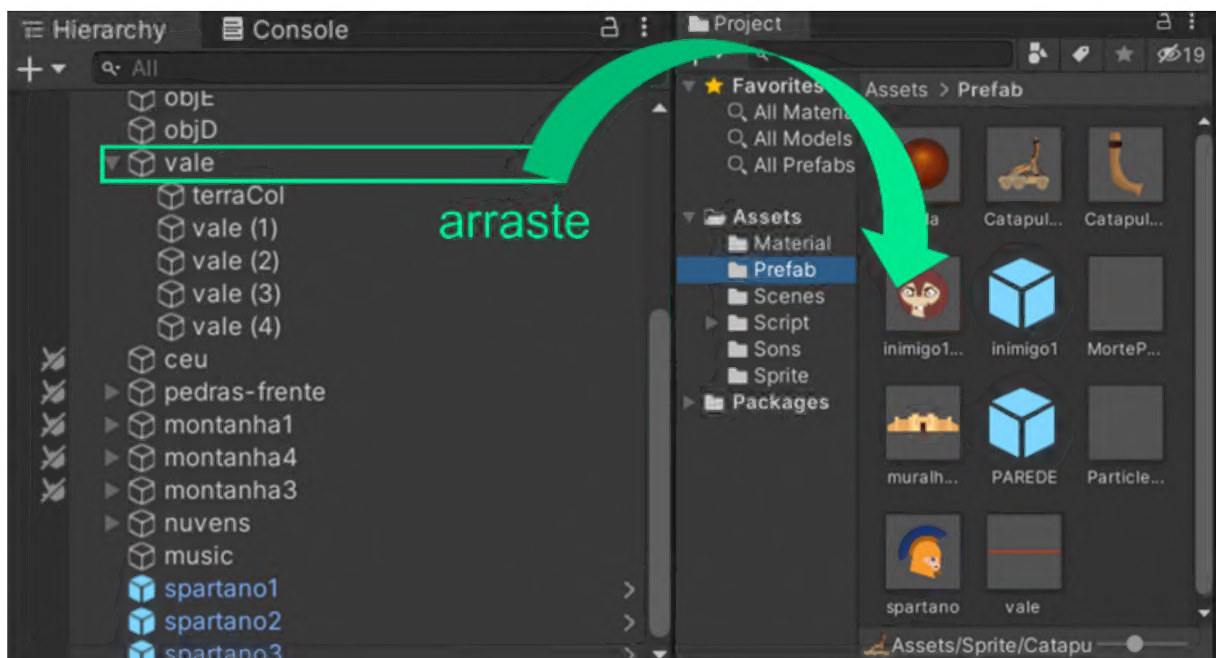
Figura 6 – Prefab Catapulta.



Fonte: Autor, 2022.

7º Passo – Arraste as sprites editadas para a pasta Prefabs para torná-las dessa categoria (Figura 7).

Figura 7 – Criando Prefab



Fonte: Autor, 2022.

Após a conclusão da Atividade transforme a pasta em uma do tipo compacta, poste o resultado e responda:

1) O que aprendeu?



2) Como aprendeu?

Gabarito Atividade Prática

1) Aprendeu a preparar os Assets colocando elementos de física 2D (Colliders, Rigibory etc.) e salvá-los como Prefabs.

2) Você teve uma aprendizagem ativa, por meio do trabalho desenvolvido, ou seja, "mão na massa".

Ir para exercício